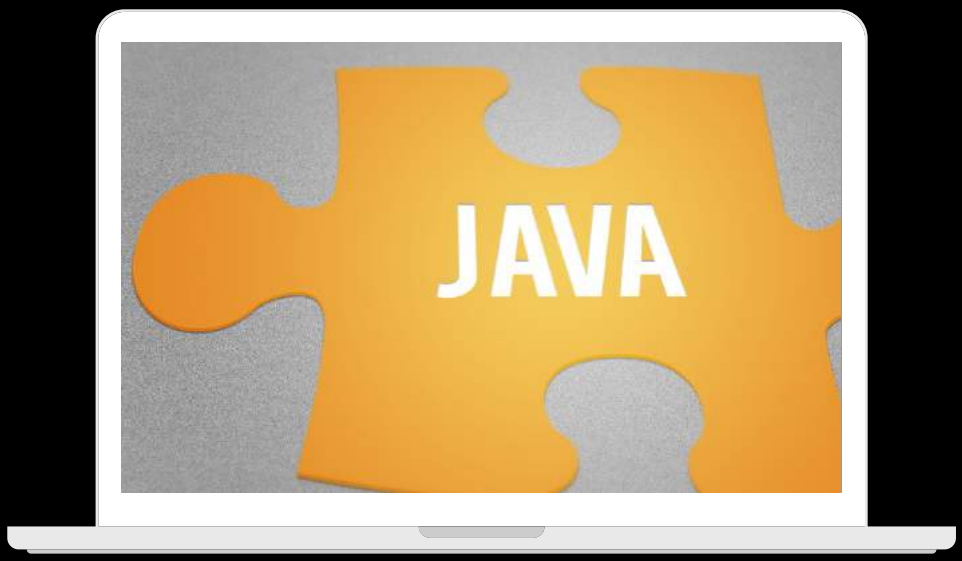
DIGITAL DOWNLOAD हिंदी

Java Programming for Beginners

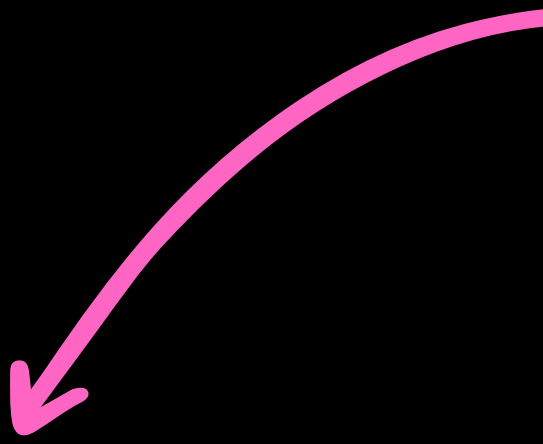
Thanks for purchasing this Digital Download.



What's the success?

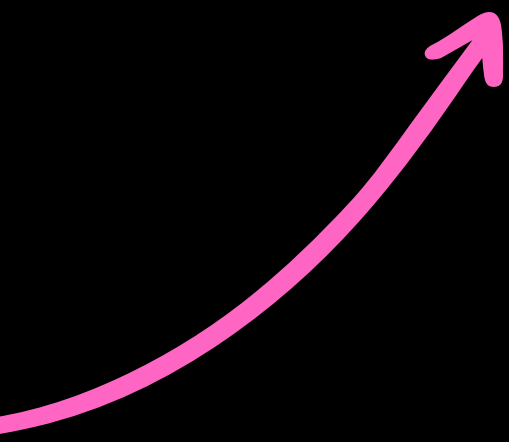
“We learn from failure, not from success!”
— Bram Stoker, Dracula

Please Keep this PDF in a safe place!



SUBSCRIBE TO FREE EBOOKS

www.cprogrammingbookinhindi.com



Java Programming for Beginners in Hindi

Copyright © 2017 by Machhindra Kalan

All rights reserved. No part of this work may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage or retrieval system, without the prior written permission of the copyright owner and the publisher.

Trademarked names may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, we use the names only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

Lead Editor: Machhindra Kalan

Distributed to the book trade worldwide by Web Study Lab Pvt Ltd, Location - Mumbai, Palava city Pin 421204

E-mail: machhi29@gmail.com, or

Visit: www.cprogrammingbookinhindi.com

For information on translations, please contact Web Study Lab Pvt Ltd, Location - Mumbai, Palava city Pin 421204

Phone +91 86555-67778

The information in this book is distributed on an “as is” basis, without warranty. Although every precaution has been taken in the preparation of this work, the author shall not have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the information contained in this book.

www.cprogrammingbookinhindi.com
3

This book
is dedicated
to those

who really wants
to be a

Java Programming for Beginners in Hindi

INDEX OF CONTENTS

Table of Contents

Chapter 1 Introduction…….
A Brief History of Computer Programming

Chapter 2 Getting Started……..
What is Programming?
What is Data?
What is Compiler?
What is interpreter?
Programming Environment Setup
Compilation and Execution

Chapter 3 Basic Programming Terms……..
Tokens
Writing Java
Separator Tokens
Operator Tokens
Literals

Chapter 4 Basic of Java Program …….
Basic Structure of Java Program
The main () Method.
Access Control
Package in java
The import Keyword
Access Modifiers

Chapter 5 Variables, Data Types and Keywords …….
Understanding Variables.
Naming Variables.
Types of Variables.
Data Types in Java.
Types
Value and Reference Types
Strong Typing
Understanding floating points.
Keywords
Return Keyword
Are Errors Bad?
Compile Time and Run Time Errors

Chapter 6 Methods and Operators ……..
What are Functions?

Parameter Lists
Side Effects
Multiple Arguments
Code Blocks
Logic and Operators

Chapter 7 Controlling Execution, Arrays and Loops …….
Controlling Execution
Loops
Arrays

Chapter 8 Object Oriented Programming ……
Classes
Introduction to Objects
Characteristics of OOP
An object has an interface
An object provides services
The hidden implementation
Reusing the implementation
Inheritance
Polymorphism

Chapter 9 Exception Handling ……..
Error Handling with Exceptions
Basic exceptions
Catching an exception
Catching any exception

Chapter 10 Algorithms and the Big O Notation ……..
Thinking in Algorithms.
The Big O Notation.

Chapter 11 Data Structures ……..
Binary Search.
Bubble Sort
Insertion Sort
Merge Sort.
Quick Sort.
Selection Sort
Linked List

Chapter 12 Network Programming …….
What is network programming?
Socket Programming
URL Processing

CHAPTER 1

INTRODUCTION

कम्प्यूटर प्रोग्रामिंग का संक्षिप्त इतिहास: कैसे प्रोग्रामिंग आया**?**

MECHANICAL COMPUTERS:

पहला प्रोग्राम कंप्यूटर यकीनन 1820 के दशक में चार्ल्स बैबेज द्वारा निर्मित बैबेज मशीन है। दसियों हज़ारों चलते भागों से बना है और कई टन वजनी है, यह एक विशाल यांत्रिक कैलकुलेटर था जो 31 अंकों की संख्या प्रदर्शित कर सकता था। इसे अंतर इंजन कहा जाता था, और 1824 में बैबेज ने आकाश के माध्यम से ग्रहों की आवाजाही के लिए इस्तेमाल की जाने वाली तालिकाओं के एक सेट को सही करने के लिए ब्रिटिश एस्ट्रोनॉमिकल सोसाइटी से स्वर्ण पदक जीता था। 1833 में, एक काउंटेस जिसे ऑगस्टा एडा किंग ऑफ लवलेस के नाम से जाना जाता है, जिसे आमतौर पर ऐडा लवलेस के रूप में जाना जाता है, ने बैबेज से एक पार्टी में मुलाकात की, जहां उन्हें अंतर इंजन से परिचित कराया गया था। कई साल बाद, यह स्पष्ट हो गया कि वह मशीन को खुद चार्ल्स से बेहतर समझ रही थी। लवेलस के अर्ल के लिए गृहिणी होने के बावजूद, उन्होंने अंतर इंजन के संचालन के साथ–साथ इसके उन्नत संस्करण, एनालिटिकल इंजन के संचालन पर कई दस्तावेज लिखे। उसे अक्सर अंतर इंजन और उसके प्रलेखन के साथ अपने काम के लिए पहला कंप्यूटर प्रोग्रामर माना जाता है।

LOGIC:

1848 में, जॉर्ज बोले ने हमें बूलियन तर्क दिया। यह अंतर इंजन और अपनी उपस्थिति बनाने के लिए पहले सामान्य प्रोग्रामेबल कंप्यूटर के बीच लगभग एक सदी लगेगा। जॉर्ज के लिए धन्यवाद, आज हमारे कंप्यूटर बाइनरी (1s और 0s) में गिने जाते हैं, और हमारा सॉफ्टवेयर सही या गलत के संदर्भ में सोचता है। 1887 में, डोर यूजीन फेल्ट ने बटन के साथ एक कंप्यूटिंग मशीन का निर्माण किया; उसके लिए धन्यवाद, हमारे कंप्यूटर में कीबोर्ड हैं। 1890 के दशक में अमेरिकी जनगणना को रिकॉर्ड करने के लिए 1s और 0s का प्रतिनिधित्व करने वाले छिद्रों के साथ एक टेबुलेटिंग मशीन ने कागज का इस्तेमाल किया। तब इसने यूएस $ 2

मिलियन और 6 साल के काम को बचाया। 1900 के दशक में, निकोला टेस्ला और अलेक्जेंडर ग्राहम बेल के बीच, आधुनिक कंप्यूटर ट्रांजिस्टर के आविष्कार के साथ आसन्न थे। 1936 में, कोनराड ज़्यूस ने Z1 कंप्यूटर बनाया, जो बैबेज की तरह का एक और मैकेनिकल कंप्यूटर था, लेकिन इस बार इसमें छेद वाले टेप का इस्तेमाल किया गया, जैसे कि टेबुलेटिंग मशीन। वह बाद में Z3 को 1941 बनाने के लिए आगे बढ़े।

COMPUTER SCIENCE:

1940 के दशक में, एक ब्रिटिश कंप्यूटर वैज्ञानिक, एलन ट्यूरिंग, ने बॉम्बे नामक एक एनक्रिप्टेड संदेश डिकोडर को डिज़ाइन और निर्मित किया, जिसने द्वितीय विश्व युद्ध के दौरान मित्र देशों की सेनाओं की मदद के लिए विद्युत स्विच का उपयोग किया। 1930 और 1950 के दशक के बीच, ट्यूरिंग ने कंप्यूटर वैज्ञानिकों को सूचित किया कि कंप्यूटर सिद्धांत रूप में, गणना योग्य कुछ भी हल कर सकते हैं, इस अवधारणा को ट्यूरिंग पूर्णता कहते हैं। ये सभी घटक हमारे आधुनिक कंप्यूटर की ओर ले जाने लगे। 1940 के मध्य में, जॉन वॉन न्यूमैन ने अपने सिद्धांत की मदद से प्रदर्शित किया कि कंप्यूटर को सरल घटकों का उपयोग करके बनाया जा सकता है। इस तरह से हार्डवेयर को नियंत्रित करने वाला सॉफ्टवेयर जटिल व्यवहार को जोड़ सकता है। टेस्ला और बेल के लिए धन्यवाद, Z3 पूरी तरह से इलेक्ट्रॉनिक भागों से बना था। इसमें गणना करते समय तर्क का पहला उपयोग शामिल था, जिससे यह पहला पूर्ण ट्यूरिंग–पूर्ण कंप्यूटर बना। 1954 में, टेक्सास इंस्ट्रूमेंट्स में गॉर्डन टील ने सिलिकॉन–आधारित ट्रांजिस्टर पेश किया, जो ठोस–राज्य इलेक्ट्रॉनिक्स के निर्माण के लिए एक प्रमुख घटक है जो आज हर कंप्यूटर में उपयोग किया जाता है।

SOFTWARE:

Z3 को प्लैंकल्कल का उपयोग करके प्रोग्राम किया गया था, पहली उच्च–स्तरीय प्रोग्रामिंग भाषा, जो कि मशीन बनाने के कुछ समय बाद ही Zuse द्वारा आविष्कार की गई थी। 1952 में ग्रेस हॉपर ने पहला संकलक, सॉफ्टवेयर बनाया जो मानव–पठनीय शब्दों को मशीन संचालन में अनुवाद कर सकता था। पहली बार, प्रोग्रामर को यह जानने की आवश्यकता नहीं थी कि कंप्यूटर चलाने के लिए सॉफ़्टवेयर बनाने के लिए हार्डवेयर में ट्रांजिस्टर कैसे संचालित होते हैं। इसने कंप्यूटर प्रोग्रामिंग को एक नए दर्शकों के लिए खोल दिया। 1950 और 1970 के दशक के बीच, कंप्यूटर और कंप्यूटर भाषा जटिलता और क्षमता के मामले में बढ़ रहे थे। 1958 में, जॉन मैकार्थी ने मैसाचुसेट्स इंस्टीट्यूट ऑफ टेक्नोलॉजी (MIT) में LISP का आविष्कार किया। सिमूला 67 में ऑब्जेक्ट ओरिएंटेड प्रोग्रामिंग 1967 दिखाई दी। इंपीरियल और प्रक्रियात्मक प्रोग्रामिंग ने पास्कल इन 1970 में अपनी उपस्थिति दर्ज की।

MODERN COMPUTER LANGUAGE:

1920 में बेल द्वारा शुरू की गई बेल लैब्स ने डेनिस रिची को 1967 काम पर रखा। 1974 में, रिची ने सी प्रोग्रामिंग भाषा प्रकाशित की, जो सबसे लोकप्रिय कंप्यूटर प्रोग्रामिंग भाषा बन गई है। 1983 में, C++ ने इसका निर्माण किया उपस्थिति के रूप में "सी कक्षाओं के साथ।" जेम्स गोसलिंग, माइक शेरिडन और पैट्रिक नौगटन ने जून 1991 में जावा भाषा परियोजना शुरू की जावा मूल रूप से इंटरैक्टिव टेलीविजन के लिए डिज़ाइन किया गया था, लेकिन यह उस समय डिजिटल केबल टेलीविजन उद्योग के लिए बहुत उन्नत था। भाषा को शुरू में ओक एक ओक के पेड़ के बाद कहा जाता था जो गोसलिंग के कार्यालय के बाहर खड़ा था। बाद में इस परियोजना के नाम से ग्रीनंड को अंततः जावा नाम दिया गया, जावा कॉफी से। गोसलिंग ने जावा को एक सी / सी –स्टाइल सिंटैक्स के साथ डिज़ाइन किया है जो सिस्टम और एप्लिकेशन प्रोग्रामर परिचित होगा।

सन माइक्रोसिस्टम्स ने जावा 1.0 के रूप में पहला Public कार्यान्वयन continue किया 1995 में यह वादा किया था "एक बार लिखें, कहीं भी दौड़ें" (श्रम), बिना किसी लागत के लोकप्रिय प्लेटफार्मों पर रन–टाइम। काफी सुरक्षित और विन्यास योग्य सुरक्षा की विशेषता, इसने नेटवर्क– और फ़ाइल–एक्सेस प्रतिबंधों की अनुमति दी। प्रमुख वेब ब्राउज़रों ने जल्द ही वेब पृष्ठों के भीतर जावा एप्लेट्स को चलाने की क्षमता को शामिल किया और जावा जल्दी लोकप्रिय हो गया। जावा 1.0 कंपाइलर को जावा में आर्थर वैन हॉफ द्वारा जावा 1.0 के साथ सख्ती से अनुपालन करने के लिए फिर से लिखा गया था भाषा विनिर्देश। जावा 2 के आगमन के साथ (शुरू में J2SE 1.2 के रूप में continue किया गया दिसंबर 1998 –1999), नए संस्करणों में विभिन्न प्रकार के प्लेटफार्मों के लिए कई कॉनफ़िगरेशन बनाए गए थे। J2EE में प्रौद्योगिकी और एपीआई शामिल थे

एंटरप्राइज़ एप्लिकेशन के लिए आमतौर पर सर्वर वातावरण में चलते हैं, जबकि J2ME में एपीआई मोबाइल अनुप्रयोगों के लिए अनुकूलित होते हैं। डेस्कटॉप संस्करण का नाम बदलकर J2SE कर दिया गया। 2006 में, विपणन उद्देश्यों के लिए, Sun ने क्रमशः जावा ईई, जावा एमई और जावा एसई के रूप में नए जे 2 संस्करणों का नाम बदला।

1997 में, सन माइक्रोसिस्टम्स ने जावा को औपचारिक रूप देने के लिए ISO / IEC JTC 1 मानकों बॉडी और बाद में Ecma International से संपर्क किया, लेकिन यह जल्द ही प्रक्रिया से हट गया। जावा एक वास्तविक तथ्य है, जिसे जावा समुदाय प्रक्रिया के माध्यम से नियंत्रित किया जाता है। एक समय में, अपने मालिकाना सॉफ्टवेयर स्थिति के बावजूद, Sun ने अपने अधिकांश जावा कार्यान्वयन बिना किसी शुल्क के उपलब्ध कराए। सन ने जावा एंटरप्राइज सिस्टम जैसे विशेष उत्पादों के लाइसेंस के माध्यम से जावा से राजस्व अर्जित किया।

13 नवंबर 2006 को, Sun ने अपने Java वर्चुअल मशीन (JVM) को GNU

जनरल पब्लिक लाइसेंस (GPL) की शर्तों के तहत मुफ्त और ओपन–सोर्स सॉफ्टवेयर, (FOSS) के रूप में continue किया। 8 मई, 2007 को, Sun ने प्रक्रिया को समाप्त कर दिया, जिससे J के सभी कोर कोड मुफ्त सॉफ्टवेयर / ओपन–सोर्स वितरण शर्तों के तहत उपलब्ध हो गए, कोड के एक छोटे से हिस्से से अलग, जिसमें Sun ने कॉपीराइट नहीं रखा था।

THE FUTURE OF COMPUTER LANGUAGES:

1920 के दशक में स्थापित एक अमेरिकी कंपनी इंटरनेशनल बिजनेस मशीन्स कॉर्पोरेशन (आईबीएम), वर्तमान में लगभग एक सदी पहले खोजे गए कण भौतिकी का उपयोग कर नई तकनीक इंजीनियरिंग कर रही है। 1920 के दशक के उत्तरार्ध में जर्मन भौतिकविदों के काम से प्राप्त आधुनिक सप्रिंट्रोनिक्स। बिना कण का फ्रेडरिक हर्मन हंड जैसे भौतिकी और भौतिक विज्ञानी, हम क्वांटम टनलिंग को नहीं समझेंगे, और सॉलिड–स्टेट डिस्क या यूनिवर्सल सीरियल बस (USB) थंब ड्राइव मौजूद नहीं होगी। की खोज और पुष्टि के बाद हिग्स बोसोन, आप पूछ सकते हैं "परेशान क्यों?" इस सवाल का एक और सौ वर्षों के लिए एक अच्छा जवाब नहीं मिल सकता है, लेकिन यह इसके लायक होगा। जावा अंतिम कंप्यूटर भाषा नहीं है; कई अन्य लोगों को हाल ही में पेश किया गया है, विशेष रूप से Google द्वारा गो नामक एक प्रविष्टि।

Microsoft, को पार करने के लिए नहीं, F# भी पेश किया है।
कई अपेक्षाकृत नई प्रोग्रामिंग भाषाएं हैं जिन्हें हाल के वर्षों में लोकप्रिय बनाया गया है। हालांकि भाषाएं अलग हैं, प्रिंसिपल अवधारणाएं सभी समान हैं। एक बार मूल विचारों को समझने के बाद, यहां प्राप्त ज्ञान को नई भाषाओं में लागू करना और सीखना continue रखना आसान है। आज क्वांटम कंप्यूटर अपनी उपस्थिति बनाना शुरू कर देते हैं। आज के ट्रांजिस्टर के साथ व्यवहार करते हुए, हम निर्धारित मूल्यों को भेजते हैं और प्राप्त करते हैं। क्वांटम कंप्यूटर 1s और 0s के साथ काम नहीं करते हैं; बल्कि वे 0 और 1 के बीच कई मान संग्रहीत करते हैं. इससे निपटने के लिए, वर्तमान में नई प्रोग्रामिंग भाषाओं को विकसित किया जा रहा है।

CHAPTER 2

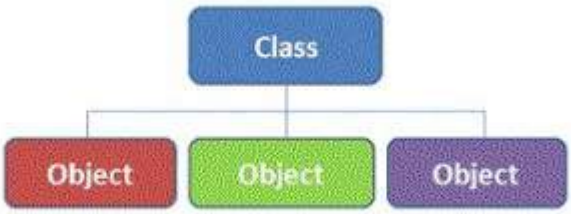
GETTING STARTED

प्रोग्रामिंग क्या है?

यह सब कोड लिखने के बारे में है। प्रोग्रामिंग एक ऐसी प्रक्रिया है जिसमें हम डेटा को व्यवस्थित करते हैं और उन डेटा के साथ कुछ करने के लिए तर्क का उपयोग करते हैं। डेटा वह सब कुछ है जो एक कंप्यूटर स्टोर कर सकता है; वे एक वीडियो गेम में संख्याओं से लेकर ज़ोंबी पात्रों तक हो सकते हैं।

आप स्रोत कोड नामक फ़ाइलों में पाठ लिखकर ऐसा करते हैं। पाठ फ़ाइलों में लिखा गया स्रोत कोड आधी सदी पहले कंप्यूटिंग मशीनों द्वारा उपयोग किए जाने वाले पंच कार्ड की जगह लेता है।

जब डेटा को तर्क के साथ जोड़ दिया जाता है और फिर एकल फ़ाइल में लिखा जाता है, तो उन्हें एक class कहा जाता है। classes भी डेटा हैं, और जैसे कि अधिक तर्क के साथ प्रबंधित किया जा सकता है।



कंप्यूटर की मेमोरी में ऑब्जेक्ट्स बनाने के लिए कक्षाओं का उपयोग किया जाता है और उनका स्वयं का जीवन बनाने के लिए डुप्लिकेट किया जा सकता है। कक्षाओं का उपयोग वस्तुओं के निर्माण के लिए किया जाता है। कक्षा के भीतर डेटा का प्रत्येक टुकड़ा उस वस्तु का एक हिस्सा बन जाता है। किसी class के अंदर डेटा के विभिन्न भाग को कक्षा के सदस्य कहा जाता है।

कक्षा के सदस्यों को भी फ़ंक्शन या methodयों नामक तर्क का हिस्सा हो सकता है।

उदाहरण के लिए, लाश के एक गिरोह के साथ एक खेल में, प्रत्येक ज़ोंबी

को एक ज़ोंबी class से नकल या उकसाया जाता है। प्रत्येक ज़ोंबी में कक्षा में प्रत्येक विशेषता या डेटा तत्व के लिए अद्वितीय मान हैं। इसका मतलब हिट पॉइंट्स हैं, और प्रत्येक डुप्लिकेट ज़ोंबी ऑब्जेक्ट के लिए स्थान अद्वितीय हैं। किसी class से निर्मित वस्तुओं को उदाहरण कहा जाता है। परिवारों के समान, वस्तुएँ एक दूसरे से गुण प्राप्त कर सकती हैं।

बच्चे को कभी–कभी एक उपclass कहा जाता है जो अपने माता–पिता से विशेषता प्राप्त करता है। उदाहरण के लिए, एक ज़ोंबी से बनाया गया बच्चा दिमाग के लिए माता–पिता की भूख का वारिस हो सकता है। उपयोगी होने के लिए, बच्चा ज़ोंबी नई वस्तुओं को भी जोड़ सकता है और अपने मूल class से विरासत में मिली वस्तुओं को बदल सकता है। नतीजतन, अब बच्चे की लाश में ऐसे जाल हो सकते हैं जो माता–पिता के पास नहीं थे। वस्तुएं घटनाओं और संदेशों के माध्यम से एक दूसरे से बात करती हैं।
लाश पर शूटिंग एक घटना बना सकते हैं, या प्रोग्रामर शब्दों में, यह "उठाता है" एक घटना। बुलेट इफेक्ट घटना ज़ोंबी class को तब बताती है जब बुलेट से हिट हो।

इवेंट्स क्लास को उसके डेटा पर एक्शन लेने के लिए कमांड करता है, जो कि फ़ंक्शंस में आता है।

फ़ंक्शंस, जिन्हें methodयों के रूप में भी जाना जाता है, तर्क के अनुभाग हैं जो डेटा पर functions करते हैं। वे आपकी कक्षा को अतिरिक्त उपक्रम बनाने और अभी तक अधिक वस्तुओं से बात करने की अनुमति देते हैं।

जैसा कि खिलाड़ी ट्रिगर दबाता है और जॉयस्टिक को चारों ओर घुमाता है, फिर भी अधिक घटनाओं को उठाया जा सकता है और संदेश भेजे जा सकते हैं। इवेंट और संदेश खिलाड़ी को आपकी दुनिया के साथ बातचीत करने की अनुमति देते हैं; तर्क घटनाओं और वस्तुओं को एक

साथ अपने खेल का निर्माण।

डेटा क्या है?

डेटा, एक सामान्य अर्थ में, संज्ञा की तरह है। संज्ञा की तरह, डेटा एक व्यक्ति, स्थान या चीज हो सकता है। प्रोग्रामर इन संज्ञाओं को वस्तुओं के रूप में संदर्भित करते हैं, और इन वस्तुओं को स्मृति में variables के रूप में संग्रहीत किया जाता है। शब्द variable कुछ ऐसा है जो बदल सकता है, लेकिन यह हमेशा ऐसा नहीं होता है। जानकारी डालने के लिए आपके कंप्यूटर की मेमोरी में एक स्थान के रूप में एक variable के बारे में सोचना बेहतर है। जब आप मैडलिब जैसे शब्द का खेल खेलते हैं तो आप किसी का नाम, कोई वस्तु, क्रिया विशेषण और कोई स्थान पूछ सकते हैं। आपका परिणाम "गार्थ एक जैकेट खा गया, और कपड़े धोने की चटाई पर खेला जा सकता है।"इस मामले में नाम, वस्तु, क्रिया विशेषण और स्थान परिवर्तनशील प्रकार हैं। डेटा वह शब्द है जिसका उपयोग आप वेरिएबल को असाइन करने के लिए करते हैं।

प्रोग्रामर शब्द प्रकार का उपयोग यह बताने के लिए करते हैं कि किस प्रकार का डेटा संग्रहीत किया जा रहा है। कंप्यूटर अंग्रेजी में धाराप्रवाह नहीं हैं और आमतौर पर अंग्रेजी प्रकार की संज्ञा और विशेषण के बीच अंतर नहीं जानते हैं, लेकिन वे अक्षरों और विभिन्न प्रकार की संख्याओं के बीच अंतर को जानते हैं। जावा या किसी अन्य भाषा में कई पूर्वनिर्धारित प्रकार हैं।

यदि आप इसे नए प्रकार के डेटा बनाने की क्षमता में जोड़ते हैं, तो हम जिस प्रकार के डेटा को स्टोर कर सकते हैं, वह व्यावहारिक रूप से असीमित है।

C# बिल्ट–इन प्रकारों को कभी–कभी POD, या सादे पुराने डेटा कहा जाता है।

POD शब्द मूल C मानक से आया है जो इसके मूल को वापस डेटिंग

के लिए 1979 पाता है. POD प्रकारों को उनके मूल कार्यान्वयन से मौलिक रूप से नहीं बदला गया है।
अब तक हमने कई बार शब्द प्रकार का उपयोग किया है। प्रोग्रामर स्टोर किए जाने वाले डेटा की विविधता का वर्णन करने के लिए शब्द प्रकार को परिभाषित करते हैं।

एक कंपाइलर क्या है?
आम तौर पर compile एक ऐसा शब्द है जिसे अक्सर हर कोई सुनता है जो दूर से भले ही प्रोग्रामिंग से जुड़ा हो। एक कंपाइलर एक सॉफ्टवेयर प्रोग्राम है जो उच्च–स्तरीय स्रोत कोड को परिवर्तित करता है जो एक डेवलपर द्वारा उच्च स्तरीय प्रोग्रामिंग भाषा में मशीन भाषा में निम्न स्तर के ऑब्जेक्ट कोड (बाइनरी कोड) में लिखा जाता है, जिसे प्रोसेसर द्वारा समझा जा सकता है। उच्च स्तरीय प्रोग्रामिंग को मशीन भाषा में परिवर्तित करने की प्रक्रिया को compile के रूप में जाना जाता है।

प्रोसेसर ऑब्जेक्ट कोड को निष्पादित करता है, जो इंगित करता है कि प्रोसेसर के अंकगणितीय तर्क इकाई में बाइनरी हाई और लो सिग्नल की आवश्यकता होती है।

आइए एक उदाहरण लेते हैं कि यह समझने के लिए कि उपरोक्त परिभाषा का क्या अर्थ है।

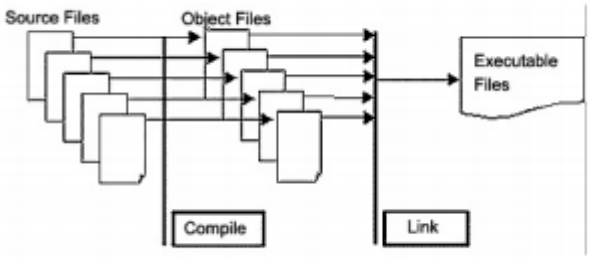
```
#include<stdio.h>
    Void main()
    {
      printf("Hello, world!");
    }
```

यह सबसे बुनियादी कार्यक्रम प्रिंट करता है या "हैलो, व र्ल्ड!" शब्दों को प्रदर्शित करता है। कंप्यूटर स्क्रीन पर। लेकिन एक समस्या है। यह उतना सरल नहीं हैं। पर्दे के पीछे बहुत abstraction जटिल चीजें चल रही हैं। हमें इन चीजों के अंदर झांकना चाहिए। कठिन सत्य यह है कि हमारा

कंप्यूटर एक स्रोत फ़ाइल (helloworld.c) में निहित आदेशों / निर्देशों को नहीं समझ सकता है।), क्योंकि C एक उच्च–स्तरीय भाषा है, जिसका अर्थ है, इसमें विभिन्न वर्ण, प्रतीक और शब्द शामिल हैं, जो जटिल, संख्या–आधारित निर्देशों का उदाहरण देते हैं। प्रिंटफ, main, हेडर फाइलें आदि। केवल एक निर्देश जो कंप्यूटर निष्पादित कर सकता है, वे मशीन भाषा में लिखे गए हैं, जिसमें पूरी तरह से संख्याएं होती हैं।

यह 0 और 1 के संदर्भ में द्विआधारी भाषा है। इससे पहले कि हमारा कंप्यूटर हमारे सी प्रोग्राम को चला सके, हमारे कंपाइलर को हमारे helloworld.c को परिवर्तित करना चाहिए। एक वस्तु फ़ाइल में तब लिंकर नामक एक प्रोग्राम को ऑब्जेक्ट फ़ाइल को एक निष्पादन योग्य फ़ाइल में बदलना चाहिए।

नीचे दी गई ड्राइंग प्रक्रिया को दर्शाती है।



एक कंपाइलर जो मशीन लैंग्वेज को हाई–लेवल नेचुरल लैंग्वेज में कन्वर्ट करता है, उसे डिकम्पॉइलर कहा जाता है। कंपाइलर्स जो किसी सिस्टम पर चलने वाले ऑब्जेक्ट कोड का उत्पादन करते हैं, क्रॉस–कंपाइलर कहलाते हैं। अंत में, एक संकलक जो एक प्रोग्रामिंग भाषा को दूसरे में परिवर्तित करता है, भाषा अनुवादक कहलाता है।

एक कंपाइलर चार प्रमुख चरणों को निष्पादित करता है:

स्कैनिंग: स्कैनर स्रोत कोड से एक समय में एक character को पढ़ता है और किस लाइन में कौन सा character मौजूद है, इस पर नज़र रखता

है।

लेक्सिकल एनालिसिस: कंपाइलर उन वर्णों के अनुक्रम को परिवर्तित करता है जो स्रोत कोड में वर्णों के तार (जिसे टोकन के रूप में जाना जाता है) की एक श्रृंखला में परिवर्तित होते हैं, जो एक विशिष्ट नियम द्वारा एक लेक्सिकल विश्लेषक द्वारा प्रोग्राम से जुड़े होते हैं। एक प्रतीक तालिका का उपयोग लेक्सिकल विश्लेषक द्वारा स्रोत कोड में शब्दों को संग्रहीत करने के लिए किया जाता है जो उत्पन्न टोकन के अनुरूप होता है।

Syntactic Analysis: इस चरण में, syntax विश्लेषण किया जाता है, जिसमें यह निर्धारित करना शामिल है कि क्या शाब्दिक विश्लेषण के दौरान बनाए गए टोकन उनके उपयोग के abstraction उचित क्रम में हैं। कीवर्ड के एक सेट का सही क्रम, जो एक वांछित परिणाम दे सकता है, को सिंटैक्स कहा जाता है। संकलक को सटीकता सुनिश्चित करने के लिए स्रोत कोड की जांच करनी होती है।

Semantic Analysis: यह कदम कई मध्यवर्ती चरणो से मिलकर बना है। सबसे पहले, टोकन की संरचना की जाँच की जाती है, साथ ही एक दिए गए भाषा में व्याकरण के संबंध में उनके आदेश के साथ। टोकन संरचना का अर्थ पार्सर और विश्लेषक द्वारा अंत में एक मध्यवर्ती कोड उत्पन्न करने के लिए व्याख्या की जाती है, जिसे ऑब्जेक्ट कोड कहा जाता है। ऑब्जेक्ट कोड में निर्देश शामिल होते हैं जो प्रोग्राम में सामना करने पर संबंधित टोकन के लिए प्रोसेसर कार्रवाई का प्रतिनिधित्व करते हैं। अंत में, पूरे कोड को पार्स किया जाता है और यह जांचने के लिए व्याख्या की जाती है कि क्या कोई अनुकूलन संभव है। एक बार अनुकूलन किए जाने के बाद, अंतिम ऑब्जेक्ट कोड उत्पन्न करने के लिए ऑब्जेक्ट कोड में उपयुक्त संशोधित टोकन डाले जाते हैं, जो किसी फ़ाइल के अंदर सहेजा जाता है।

तो एक कंपाइलर एक प्रोग्राम है जो इनपुट के रूप में एक प्रोग्राम (कुछ उच्च–स्तरीय प्रोग्रामिंग भाषा में) और आउटपुट मशीन भाषा कोड (कुछ मशीन आर्किटेक्चर के लिए) पढ़ता है। मशीन भाषा कोड बाद में हर बार अलग–अलग इनपुट डेटा का उपयोग करके किसी भी समय निष्पादित किया जा सकता है। एक दूसरे उदाहरण के रूप में, java संकलक (compiler) javac एक रूपांतरित करता है। a **.java** स्रोत फ़ाइल में पदजमतचतमजमत फाइल जो जावा बाइटकोड में लिखी गई है, जो कि जावा वर्चुअल मशीन के रूप में जानी जाने वाली काल्पनिक मशीन के लिए मशीन भाषा है।

Interpreter क्या है?

एक interpreter एक प्रोग्राम है जो प्रोग्राम के लिए डेटा के साथ–साथ स्रोत प्रोग्राम के इनपुट के रूप में पढ़ता है, और निर्देश के बिना स्रोत प्रोग्राम निर्देश का अनुवाद करता है। उदाहरण के लिए, जावा interpreter javatranslate a .class फ़ाइल जिसे मूल रूप से अंतर्निहित मशीन पर निष्पादित किया जा सकता है।

जावा आमतौर पर compile के बजाय व्याख्या क्यों करता है?

compile का main लाभ यह है कि आप कच्ची मशीन भाषा कोड के साथ समाप्त होते हैं जिसे आपकी मशीन पर कुशलता से निष्पादित किया जा सकता है। हालांकि, यह केवल एक प्रकार की मशीन वास्तुकला (इंटेल पेंटियम, पावरपीसी) पर निष्पादित किया जा सकता है। जावा बाइटकोड जैसी इंटरमीडिएट भाषा के compile का एक प्राथमिक लाभ और फिर व्याख्या करना है कि आप प्लेटफ़ॉर्म स्वतंत्रता प्राप्त कर सकते हैं: आप उसी की व्याख्या कर सकते हैं।विभिन्न प्रकार के मशीन आर्किटेक्चर पर क्लास फ़ाइल। हालाँकि, बाईटेकोड की व्याख्या आमतौर पर पूर्व संकलित मशीन भाषा कोड को निष्पादित करने की तुलना में धीमी है। जावा बाइटकोड का उपयोग करने का एक दूसरा लाभ यह है कि यह आपके

कंप्यूटर और प्रोग्राम के बीच बफर के रूप में functions करता है। यह आपको इंटरनेट से एक अविश्वसनीय उपक्रम डाउनलोड करने और कुछ आश्वासनों के साथ अपनी मशीन पर निष्पादित करने में सक्षम बनाता है। चूंकि आप जावा interpreter (और कच्ची मशीन भाषा कोड नहीं) चला रहे हैं, इसलिए आपको सुरक्षा की एक परत द्वारा संरक्षित किया जाता है, जो दुर्भावनापूर्ण उपक्रमों के लिए सुरक्षा करता है। यह जावा और जावा बाइटकोड का संयोजन है जो एक मंच–स्वतंत्र और सुरक्षित वातावरण का उत्पादन करता है, जबकि अभी भी आधुनिक प्रोग्रामिंग abstraction का एक पूरा सेट गले लगा रहा है।

जावा बाइटकोड और जावा interpreter जावा प्रोग्रामिंग भाषा के स्वाभाविक रूप से विशिष्ट नहीं हैं। उदाहरण के लिए, आप पायथन प्रोग्रामिंग भाषा से जावा बाइटकोड में compile करने के लिए जेथॉन का उपयोग कर सकते हैं, और फिर इसकी व्याख्या करने के लिए जावा का उपयोग कर सकते हैं। इसी तरह के ML, लिस्प और फोरट्रान कंपाइलर हैं जो Java बायटेकोड में संकलित हैं। आप यूनिक्स प्रोग्राम gcj का उपयोग सीधे a से compile करने के लिए भी कर सकते हैं। .javasource फ़ाइल एक मशीन निष्पादन योग्य फ़ाइल में a.out, जो किसी भी Sparc माइक्रोप्रोसेसर पर मूल रूप से चलाया जा सकता है। इसके अतिरिक्त, आप हार्डवेयर डिज़ाइन कर सकते हैं जिसकी मशीन भाषा जावा बाइटकोड है। सन माइक्रोसिस्टम्स ने ठीक यही किया है, जिससे जावा वर्चुअल मशीन ऐसा नहीं है।

जावा बाइटकोड के बजाय एक असली मशीन भाषा का उपयोग क्यों नहीं करें?

जावा बाइटकोड एक विशिष्ट उच्च–स्तरीय प्रोग्रामिंग भाषा की तुलना में बहुत सरल है। एक नए प्रकार के कंप्यूटर के लिए एक पूर्ण जावा संकलक लिखने की तुलना में जावा बाइटकोड interpreter लिखना बहुत आसान है।

प्रोग्रामिंग **Environment** सेटअप

जब हम **Environment** सेटअप कहते हैं, तो इसका मतलब है कि हम अपनी प्रोग्रामिंग कर सकते हैं। इस प्रकार, हमें आवश्यक सॉफ्टवेयर सेटअप, यानी, हमारे पीसी पर इंस्टॉलेशन करना होगा, जिसका उपयोग कंप्यूटर प्रोग्राम लिखने, compile करने और उन्हें निष्पादित करने के लिए किया जाएगा। उदाहरण के लिए, यदि आपको इंटरनेट ब्राउज़ करने की आवश्यकता है, तो आपको अपनी मशीन पर निम्न सेटअप की आवश्यकता है:

- इंटरनेट से जुड़ने के लिए एक चरशील इंटरनेट कनेक्शन
- एक वेब ब्राउज़र जैसे कि इंटरनेट एक्सप्लोरर, क्रोम, सफारी, आदि।

इसी तरह, आपको किसी भी प्रोग्रामिंग भाषा का उपयोग करके प्रोग्रामिंग के साथ शुरू करने के लिए निम्नलिखित सेटअप की आवश्यकता होगी।

- कंप्यूटर प्रोग्राम बनाने के लिए एक टेक्स्ट एडिटर।
- बाइनरी प्रारूप में चरनक्रमों को संकलित करने के लिए एक कंपाइलर।
- सीधे चरनक्रमों को निष्पादित करने के लिए एक interpreter।

हम आपको IDE (इंटीग्रेटेड डेवलपमेंट एनवायरनमेंट) का उपयोग करने की सलाह देते हैं, सॉफ्टवेयर का आसान, बांका टुकड़ा जो टेक्स्ट एडिटर, डीबगर और कंपाइलर के रूप में काम करता है।

जब तक आप एक टेक्स्ट एडिटर और कमांड प्रॉम्प्ट विंडो में काम करने के लिए प्रतिबद्ध नहीं होते हैं (और वास्तव में ऐसा करने के लिए आकर्षक कारण हैं) आप एक अच्छी आईडीई चुनने के लिए कुछ सलाह की तलाश कर सकते हैं, विभिन्न किस्मों के पेशेवरों और विपक्षों में, किसी विशेष

आईडीई को चलाने की सापेक्ष लागत (वित्तीय या सिस्टम संसाधन), अन्य भाषाएं जो आईडीई अच्छी तरह से संभाल सकती हैं, वह ऑपरेटिंग सिस्टम (ओं) पर चलती है और कभी–कभी बहुत अधिक।

यहां प्रोग्रामर्स द्वारा उपयोग की जाने वाली दो सबसे सामान्य आईडीई की सूची दी गई है।

Eclipse

भाषाएँ: सी, सी, पायथन, पर्ल, पीएचपी, जावा, रूबी और अधिक

मूल्य: मुफ़्त

Eclipse मुक्त और मुक्त–स्रोत संपादक है, जिस पर कई विकास ढांचे आधारित हैं। यह अपने क्षेत्र में पोतियों में से एक है और कई पेशेवर डेवलपर द्वारा अत्यधिक अनुशंसित है। Eclipse एक जावा विकास वातावरण के रूप में शुरू हुआ और हल्के प्लगइन्स की एक प्रणाली के माध्यम से बहुत विस्तार किया है।

NetBeans

भाषाएँ: जावा, जावास्क्रिप्ट, पीएचपी, पायथन, रूबी, सी, सी और अधिक

मूल्य: मुफ़्त

NetBeans इस श्रेणी में सबसे अधिक अनुशंसित IDE के रूप में Eclipse के साथ साथ बराबरा है। यह मुफ़्त और खुला–स्रोत है, हर समय आने वाले अधिक प्लगइन्स के साथ कई भाषाओं का समर्थन करता है, और एक शुरुआत के लिए भी इसे स्थापित करने और उपयोग करने के लिए अविश्वसनीय रूप से सरल है।

Compile और निष्पादन **(Compilation and Execution)**

एक सरल जावा परियोजना में एक एकल निर्देशिका होती है जिसके अंदर सभी जावा स्रोत फ़ाइलें संग्रहीत होती हैं। जावा फाइलें आमतौर पर सीधे स्रोत निर्देशिका के अंदर संग्रहीत नहीं की जाती हैं, लेकिन उपनिर्देशिका में उनके पैकेज संरचना का मिलान होता है। पैकेज सिर्फ स्रोत फ़ाइलों को एक साथ समूहित करने का एक तरीका है जो एक साथ होते हैं।

जब आप स्रोत में सभी स्रोत कोड संकलित करते हैं, तो संकलक एक का उत्पादन करता है। प्रत्येक के लिए .class .java फ़ाइल। class में संकलित संस्करण .javafile है। .java फ़ाइल के लिए बाइट कोड दूसरे शब्दों में जावा फ़ाइल यह है ।

जावा वर्चुअल मशीन को निष्पादित करने वाली .class फाइलें है नही कि .java फ़ाइल है। इसलिए इसे अलग करना सामान्य है। .java फ़ाइलों से .class की फाइलें। यह सामान्य रूप से कंपाइलर को लिखने का निर्देश देकर किया जाता है। एक अलग निर्देशिका में class फ़ाइलों। इस निर्देशिका को अक्सर classes कहा जाता है, लेकिन फिर से, यह एक आवश्यकता नहीं है, और यह इस बात पर निर्भर करता है कि आप किस टूल या आईडीई आदि का उपयोग कर रहे हैं।

जावा सोर्स कोड का **compile:**

आप अपने आईडीई से सीधे अपने जावा स्रोत को संकलित कर सकते हैं (यदि आप आईडीई का उपयोग करते हैं)। या, आप जावा एसडीके के साथ आने वाले जावा कंपाइलर का उपयोग कर सकते हैं। जावा कंपाइलर के साथ कमांड लाइन से अपने जावा स्रोत कोड को संकलित करने के लिए, यह करें:

- एक कमांड प्रॉम्प्ट खोलें

- अपनी परियोजना की रूट निर्देशिका में निर्देशिका बदलें (स्रोत निर्देशिका नहीं)
- सुनिश्चित करें कि परियोजना रूट निर्देशिका में एक स्रोत निर्देशिका और एक वर्ग निर्देशिका है
- नीचे कमांड में टाइप करें (विंडोज पर – अन्य OS'es हालांकि समान दिखेंगे):

```
"c:\Program Files\Java\jdk1.8.0_25\bin\javac"src/myfirstapp/*.java -d classes
```

यह कमांड javac कमांड (जावा संकलक (**compiler**)) को निष्पादित करता है जो निर्देशिका src / myfirstapp में जावा स्रोतों को संकलित करता है। *.java का अर्थ है कि संकलक को दी गई निर्देशिका में सभी फाइलों को संकलित करना चाहिए।

Myfirstapp डायरेक्टरी रूट सोर्स डायरेक्टरी src के तहत एक पैकेज है। यदि आपके पास रूट स्रोत निर्देशिका के तहत कई पैकेज हैं, तो आपको जावा कंपाइलर को कई बार चलाना होगा। एक जावा आईडीई आपके लिए यह स्वचालित रूप से संभालता है। तो Ant, मावेन या ग्रैडल जैसे उपकरणों का निर्माण करता है।

संकलित जावा कोड चल रहा है:
एक बार कंपाइलर ने अपना काम कर लिया, तो क्लासेस डायरेक्टरी में संकलित .class की फाइलें होगा। स्रोत निर्देशिका से पैकेज संरचना (निर्देशिका संरचना) को कक्षा निर्देशिका के अंतर्गत संरक्षित किया जाएगा। आप इनमें से कोई भी चला सकते हैं। .class फाइलें जिनमें एक main () method है। आप दौड़ सकते हैं। अपने जावा आईडीई के अंदर से या कमांड लाइन से कक्षा। कमांड लाइन से यह इस तरह दिखता है:

```
"c:\Program Files\Java\jdk1.8.0_25\bin\java" -cp classes myfirstapp.MyJavaApp
```

–Cp क्लासेस फ्लैग जावा वर्चुअल मशीन को बताता है कि आपकी सभी classes कक्षाओं नामक निर्देशिका के तहत स्थित हैं। इसे "क्लास पाथ" भी कहा जाता है। (इसलिए संक्षिप्त नाम cp)

चलाने के लिए क्लास का नाम उपरोक्त कमांड में अंतिम तर्क है – **myfirstapp.MyJavaApp** हिस्सा है।

जेवीएम को यह निर्धारित करने के लिए पूरी तरह से योग्य class नाम (सभी संकुल प्लस class नाम) को जानने की आवश्यकता है कि वह कहां है। .class फ़ाइल स्थित है।

जब आप क्लास चलाते हैं तो आपकी कमांड लाइन इसी तरह की होगी (जावा ऐप से आउटपुट सहित):

```
D:\data\projects\my-first-java-app>"c:\Program
Files\Java\jdk1.8.0_25\bin\java"
-cp classes myfirstapp.MyJavaApp
Hello World!
D:\data\projects\my-first-java-app>
```

(ध्यान दें, पहले कमांड में लाइन ब्रेक नहीं होना चाहिए। मैंने इसे केवल पढ़ने में आसान बनाने के लिए रखा है।

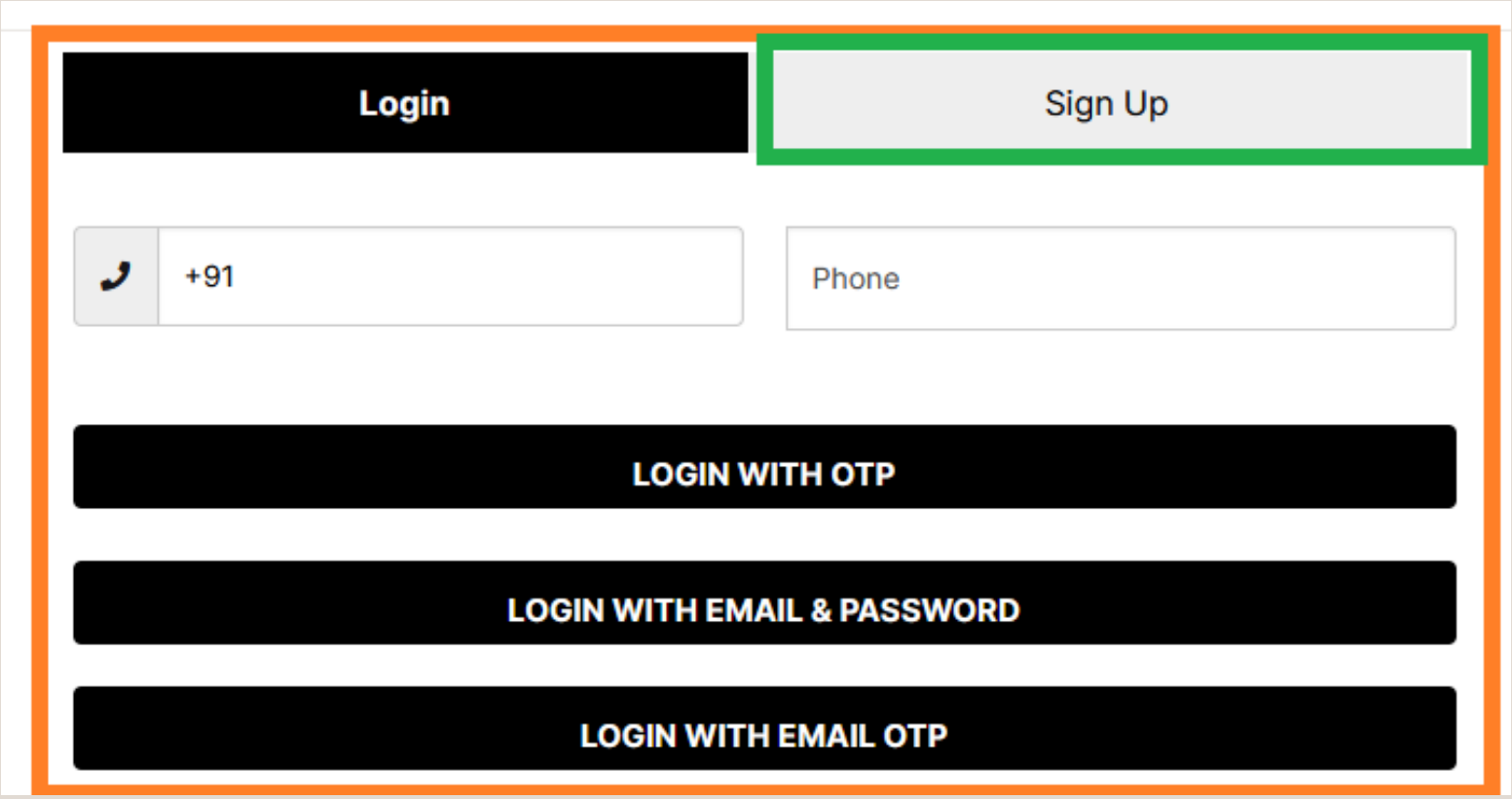
बस इतना ही! अब आप जानते हैं कि अपने जावा ऐप्स को कैसे संकलित और निष्पादित करना है! जैसा कि पहले उल्लेख किया गया है, यह एक आईडीई का उपयोग करना आसान हो सकता है, लेकिन आपके पास कोड को निष्पादित करने के लिए हर जगह आईडीई नहीं हो सकता है।

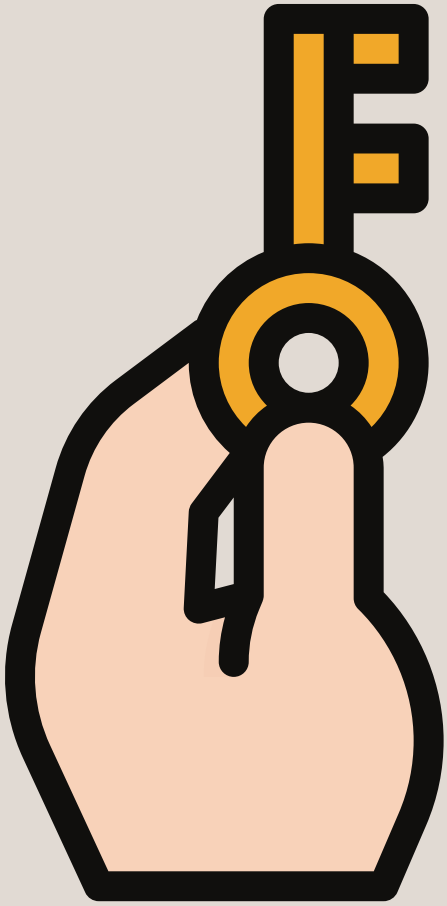
HOW TO GET COMPLETE PDF EBOOK

Pay Paytm

कोई भी **2 Ebooks** खरीदने पर **20%** डिस्काउंट- **India20**

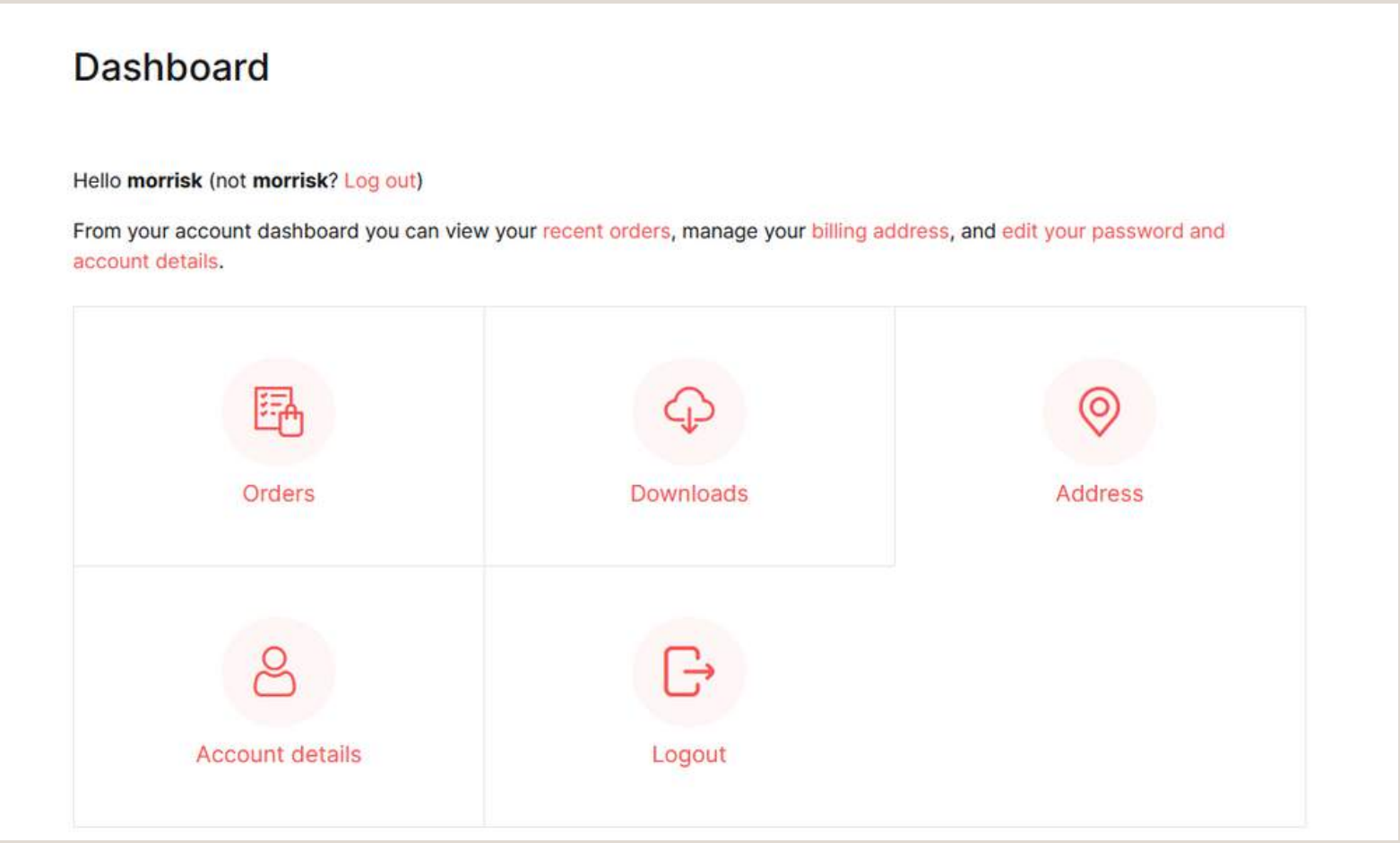
Buy Now पर क्लिक करने पर आपको **चेकआउट पेज** पर लॉगिन फॉर्म पर पहिले ईमेल से लॉगिन करना होगा।





Login Succesfull के बाद आपको पेमेंट करके Ebooks वही से डाउनलोड कर सकते है।

उसके बाद आप **My Account** पेज पर जाकर **Download** में Ebooks डाउनलोड कर सकते है।



Thank you